DO

DO ;

DO *index name* = *start value* TO *end value* [BY *increment*] ;

or

DO *index* = *start* , *end* [, *increment*] ;

Function:

DO specifies the beginning of a loop or grouped set of statements. The loop or group of statements **must** be terminated by an ENDDO ; statement.

Usage:

The first form of the DO statement (without arguments) is primarily used to specify the beginning of a block of statements which form a THEN or ELSE clause after an IF statement.

The other form of the DO statement specifies a conventional loop as in many programming languages. TSP executes the statements between the DO ... and ENDDO statement repetitively as many times as specified by the information given on the DO statement. The index or counter variable is set equal to the start value the first time through, and is changed each time through by the increment until the end value has been reached or exceeded. This test is done at the end of the loop, so the program always goes through once. DO loops can be nested with other DO loops or with DOT loops.

The start value, end value, and increment may be any real numbers (positive or negative), unlike some earlier versions of TSP which allowed only integers. If the increment is negative, obviously the index will be decremented by its absolute value, so the start value should be bigger than the end value.

The index variable is updated every time through the loop, so it may be used in computations or as a subscript. However, the DO loop in TSP is not a very efficient procedure, so that you should be wary of doing a substantial amount of variable transformation or computation with large DO loops. If you want the accumulated sum of a series, use a dynamic GENR -- for example,

```
        ACSUM = X;
        SMPL 2,N;
        ACSUM = ACSUM(-1) + X; .

or      MSD(NOPRINT) X;      (The result is in @SUM)

or      INPROD X C SUM;
```

Loops with IFs are best done with logical expressions on the right hand side of a GENR, or with a SMPLIF.

DO

Examples:

```
DO I = 2 TO 7 BY 1 ;
   SET IM1 = I-1;
   SET X(I) = X(IM1) + X(I) ;
ENDDO ;
```

```
DO I = 2 TO 7 ;
   SET IM1 = I-1 ;
   SET X(I) = X(IM1) + X(I) ;
ENDDO ;
```

The first two examples here have the same effect, since the default value of the increment is one.

```
OLSQ Y C X1 X2 ;
IF ABS(@DW-2)>.5 ; THEN ; DO ;
   AR1 Y C X1 X2 ;
   FORM EQ1 ;
ENDDO ;
ELSE ; FORM EQ1 ;
```

This example runs OLS on an equation, checks the Durbin-Watson, and runs AR1 on the same equation if the Durbin-Watson is sufficiently different from two. The DO ... ENDDO statements bracket the set of statements which are to be executed if the Durbin-Watson test fails.